

```

1 // exponentiate_matrix.cpp
2 //
3 #ifdef _WIN32
4     #pragma warning(disable:4996)
5     #include <tchar.h>
6     #include <windows.h>
7     #include <conio.h>
8     #include <direct.h>
9 #elif (defined __linux__) || (defined _AIX) || (defined __APPLE__)
10    #include <stdlib.h>
11    #include <sys/types.h>
12    #include <sys/stat.h>
13    #include <unistd.h>
14    typedef char _TCHAR;
15    #define _tmain main
16 #endif
17
18 #include <stdio.h>
19 #include <string.h>
20 #include <iostream>
21 using namespace std;
22
23 #define MAXLINE 255
24 #define MAXLEN 20
25
26 void my_getch();
27 void NacitajMaticu(char*, unsigned&, unsigned&, float[][20]);
28 void VynasobMatice(const unsigned, const unsigned, const unsigned,
29     const float[][20], const float[][20], float[][20]);
30 void UmocniMaticu(const unsigned, const unsigned, const float[][20], float[][20]);
31 void ZapisMaticu(FILE*, const unsigned, const unsigned, const unsigned,
32     const float[][20], const float[][20]);
33 void RovnaSa(const unsigned, const unsigned, const float[][20], float[][20]);
34 void GetNamesOfIOFiles(const char[], char[], const char[], char[]);
35
36 int _tmain(int argc, _TCHAR* argv[])
37 {
38     float A[MAXLEN][MAXLEN], B[MAXLEN][MAXLEN];
39     unsigned m, n, Exponent;
40     char input_file[MAXLINE], output_file[MAXLINE];
41     FILE* out;
42
43     cout << "\n Tento program nacita z textoveho suboru stvorcovu maticu A\n"
44         << "a z klavesnice prirodzene cislo k a na obrazovku a do textoveho \n"
45         << "suboru vypise k-tu mocninu matice A.";
46
47     GetNamesOfIOFiles("MATICA.TXT", input_file, "VYSLEDOK.TXT", output_file);
48     NacitajMaticu(input_file, m, n, A);
49     cout << "\n\nZadajte prirodzene cislo ako exponent matice A!\nExponent=";
50     cin >> Exponent;
51
52     UmocniMaticu(Exponent, n, A, B);
53
54     ZapisMaticu(stdout, m, n, Exponent, A, B);
55     out = fopen(output_file, "w");
56     ZapisMaticu(out, m, n, Exponent, A, B);
57     fclose(out);
58
59     my_getch();
60     return 0;
61 }
62 //-----
63 void my_getch()
64 {
65     #ifdef _WIN32
66         _getch();
67     #else
68         cout << endl;
69     #endif
70 }
71 //-----
72 void NacitajMaticu(char* MenoSuboru, unsigned& m, unsigned& n, float X[][20])
73 {

```

```

74 FILE* in;
75
76 if ((in = fopen(MenoSuboru, "r")) == NULL) {
77     cout << "\n\n Subor MATICA.TXT sa nepodarilo otvorit.\n";
78     my_getch();
79     exit(1);
80 }
81 fscanf(in, "%u%u", &m, &n);
82 for (unsigned i = 0; i < m; i++)
83     for (unsigned j = 0; j < n; j++)
84         fscanf(in, "%f", &X[i][j]);
85 fclose(in);
86 }
87 //-----
88 void VynasobMatice(const unsigned m, const unsigned n, const unsigned o,
89     const float X[][20], const float Y[][20], float Z[][20])
90 {
91     for (unsigned i = 0; i < m; i++)
92         for (unsigned j = 0; j < o; j++) {
93             float Suma = 0;
94             for (unsigned k = 0; k < n; k++)
95                 Suma += X[i][k] * Y[k][j];
96             Z[i][j] = Suma;
97         }
98 }
99 //-----
100 void UmocniMaticu(const unsigned k, const unsigned n, const float X[][20],
101     float Y[][20])
102 {
103     unsigned kk = k;
104     float Pom[20][20];
105
106     switch (kk) {
107     case 0:
108         for (unsigned i = 0; i < n; i++) {
109             for (unsigned j = 0; j < n; j++)
110                 Y[i][j] = 0;
111             Y[i][i] = 1;
112         }
113         break;
114     case 1: RovnaSa(n, n, X, Y);
115         break;
116     default: RovnaSa(n, n, X, Pom);
117         kk--;
118         while (kk--) {
119             VynasobMatice(n, n, n, X, Pom, Y);
120             RovnaSa(n, n, Y, Pom);
121         }
122     }
123 }
124 //-----
125 void ZapisMaticu(FILE* out, const unsigned m, const unsigned n, const unsigned k,
126     const float X[][20], const float Y[][20])
127 {
128     unsigned i, j;
129
130     fprintf(out, "\nZ textoveho suboru sa nacitala matica A:\n");
131     for (i = 0; i < m; i++) {
132         fprintf(out, "\n ");
133         for (j = 0; j < n; j++)
134             fprintf(out, "%10.2f", X[i][j]);
135     }
136     fprintf(out, "\n");
137     fprintf(out, "\n%u-ta mocnina matice A je:\n", k);
138     for (i = 0; i < m; i++) {
139         fprintf(out, "\n ");
140         for (j = 0; j < n; j++)
141             fprintf(out, "%10.2f", Y[i][j]);
142     }
143     fprintf(out, "\n");
144 }
145 //-----
146 void RovnaSa(const unsigned m, const unsigned n, const float A[][20], float B[][20])

```

```

147 {
148     for (unsigned i = 0; i < m; i++)
149         for (unsigned j = 0; j < n; j++)
150             B[i][j] = A[i][j];
151 }
152 //-----
153 //-----
154 void GetNamesOfIOFiles(const char name_of_input_file[], char path_to_input_file[],
155                        const char name_of_output_file[], char path_to_output_file[])
156 {
157     char current_path[MAXLINE];
158
159 #ifdef _WIN32
160     TCHAR exePath[MAXLINE];
161
162     HMODULE hModule = GetModuleHandle(NULL);
163     if (hModule != NULL) {
164         if (!GetModuleFileName(hModule, exePath, MAXLINE)) {
165             cout << "Nepodarila sa zistit cesta k exe-suboru.\n";
166             my_getch();
167             exit(1);
168         }
169     }
170     else {
171         cout << "Module handle is NULL.\n" << endl;
172         my_getch();
173         exit(1);
174     }
175
176     int iii;
177     bool flag = false;
178     for (iii = (int)wcslen(exePath); iii >= 0; iii--) {
179         if (!flag && exePath[iii] == '\\') {
180             current_path[iii + 1] = '\\0';
181             flag = true;
182         }
183         if (flag)
184             current_path[iii] = (char)exePath[iii];
185     }
186 #elif (defined __linux__ || (defined __APPLE__))
187     unsigned iii;
188     char line[MAXLINE];
189     FILE* fp;
190     if ((fp = popen("/bin/pwd", "r")) == NULL) {
191         perror("popen error");
192         exit(1);
193     }
194     if (fgets(line, MAXLINE, fp) == NULL) {
195         perror("fgets error");
196         exit(1);
197     }
198     pclose(fp);
199
200     iii = 0;
201     while (line[iii] != '\\r' && line[iii] != '\\n') {
202         current_path[iii] = line[iii];
203         iii++;
204     }
205     current_path[iii] = '\\0';
206 #elif (defined _AIX)
207     unsigned iii;
208     char line[MAXLINE];
209     FILE* fp;
210     if ((fp = popen("user/bin/pwd", "r")) == NULL) {
211         perror("popen error");
212         exit(1);
213     }
214     if (fgets(line, MAXLINE, fp) == NULL) {
215         perror("fgets error");
216         exit(1);
217     }
218     pclose(fp);
219

```

```
220     iii = 0;
221     while (line[iii] != '\r' && line[iii] != '\n') {
222         current_path[iii] = line[iii];
223         iii++;
224     }
225     current_path[iiii] = '\0';
226 #endif
227
228     path_to_input_file[0] = '\0';
229     strcat(path_to_input_file, current_path);
230 #if (defined __linux__) || (defined _AIX) || (defined __APPLE__)
231     strcat(path_to_input_file, "/inputs/");
232 #elif (defined _WIN32)
233     strcat(path_to_input_file, "inputs\\");
234 #endif
235     strcat(path_to_input_file, name_of_input_file);
236
237     path_to_output_file[0] = '\0';
238     strcat(path_to_output_file, current_path);
239 #if (defined __linux__) || (defined _AIX) || (defined __APPLE__)
240     struct stat st = { 0 };
241     strcat(path_to_output_file, "/outputs/");
242
243     if (stat(path_to_output_file, &st) == -1)
244         mkdir(path_to_output_file, 0755);
245 #elif (defined _WIN32)
246     strcat(path_to_output_file, "outputs\\");
247     if (_mkdir(path_to_output_file) != 0)
248         if (errno == ENOENT) {
249             perror("_mkdir error");
250             exit(1);
251         }
252 #endif
253     strcat(path_to_output_file, name_of_output_file);
254 }
255 //-----
256
```